sevenstax Telnet Server

Specification and User Manual

Revision No.:	1.3
State:	Released
Author:	sevenstax GmbH

Initial version:	21/12/07
Last change:	12/02/08
Last Review:	12/02/08
Publication:	Public
Filename:	sevenstaxTelnet_UserManual_v1_3

Copyright (c) 2008 by sevenstax GmbH
This document is an intellectual property of sevenstax GmbH. Unauthorized copying and distribution is prohibited.

Table of Contents

1 TELNET Server Protocol API 3

2 Using Telnet Server 3

2.1 Initialization 3
2.2 Start Telnet Server 3
2.3 Common use 4

3 Public Functions 4

3.1 Telnet 4
3.1.1 stxTELNETServ_Init (handler) 4
3.1.2 stxTELNETServ_Tick ( ) 4
3.1.3 stxTELNETServ_Start ( ) 5
3.1.4 stxTELNETServ_Stop (connID) 5
3.1.5 stxTELNETServ_Receive (data, size, connID) 5
3.1.6 stxTELNETServ_TransportLayerEvent (ev, connid) 6
3.2 NVT 6
3.2.1 stxNVT_Tick ( ) 6
3.2.2 stxNVT_Send (sDataTx, size) 6
3.2.3 stxNVT_DoEcho (value) 7
3.2.4 stxNVT_DoStringMode (value) 7

4 Implementing User Functions 8

4.1 Developing User functions 8

5 Notifycodes 10

5.1 NC_TELNET_OPEN (0, connID) 10
5.2 NC_TELNET_CONNECTED (0, connID) 10
5.3 NC_TELNET_TX (0, connID) 10
5.4 NC_TELNET_RX (0, connID) 10
5.5 NC_TELNET_OPTION_ON (option, connID) 10
5.6 NC_TELNET_OPTION_OFF (option, connID) 11
5.7 NC_TELNET_ERROR (errCode, connID) 11

6 Adjustable Parameters 11

7 Restrictions..... 13

8 Change History..... 14

1 TELNET Server Protocol API

Telnet is a network protocol used on the Internet or local area network connections. It may be enabled by *#define TELNET_SUPPORTED 1* and *#define NVT_SUPPORTED 1* in features.h.

The Telnet protocol works on top of TCP, so *#define TCP_SUPPORTED 1* must be set, too.

2 Using Telnet Server

The Telnet Server is used to connect with the embedded device in order to get information from it. For example you can get information about the network connection.
It is also possible to implement own functions to get or set variables.

The Telnet Server listens by default at TCP port 23 for Telnet Clients.

To use Telnet with sevenstaxTCP you should proceed the following way (please refer to following chapters 3...5 for details):

2.1 Initialization

1. *#define TELNET_SUPPORTED 1* and *#define NVT_SUPPORTED 1* in 'features.h'. Also TCP_SUPPORTED must be enabled there.
2. Call *stxTELNETServ_Init()* and *stxNVT_Init()* once to set-up sevenstaxTELNET's internal state.
3. In your main loop or operating system task, call *stxTELNETServ_Tick()* and *stxNVT_Tick()* repeatedly to keep alive.
4. The application has to handle the TCP notify codes for the Telnet session. So the application has to call the *stxTELNETServ_TransportLayerEvent()* when it is necessary. For example when a connection is established or when a connection was closed or a timeout occurred. Also *stxTELNETServ_Receive()* must be called if data for Telnet was received.

2.2 Start Telnet Server

1. To start the Telnet Server call: *stxTELNETServ_Start(*) and receive a Connection-ID as return value. This Connection-ID must be stored to handle Notify-Codes from TCP callback function.
2. Now the Module is able to get connected from clients on the defined Port (default: 23).

2.3 Common use

1. The device is now able to get connected from Telnet Clients. When a client is connected to the server, the client must first authenticate. After the Client is logged

in, functions can be called by commands. A list of possible commands can be displayed by typing “help” at the prompt.

2. The application has to react to the receiving Notify Codes in the handler-function.

3 Public Functions

3.1 Telnet

3.1.1 stxTELNETServ_Init (handler)

Description

Set up some internal variables and registers the application callback function. It must be called once before using any other TELNET functions.

Parameter

PROTOCOL_NOTIFY_HANDLER handler
Application handler function to receive notify codes from sevenstaxTCPs Telnet module. The handler function will be called to inform the application about received options or if an error occurred.

Return Value

void

Comment

-

3.1.2 stxTELNETServ_Tick ( )

Description

This function keeps Telnet alive. It should repeatedly be called by the application, like all the other sevenstax “Tick” functions, independently of Telnet is stopped.

Parameter

void

Return Value

void

Comment

-

3.1.3 stxTELNETServ_Start ( )

Description

This function starts the Telnet Server.

Parameter

void

Return Value

UINT16_stx Connection ID
The Connection-ID will be get from TCP Module, while making a listen at the Telnet-Server-Port.

Comment

stxTELNET_Init() must have been called before using this function.

3.1.4 stxTELNETServ_Stop (connID)

Description

This function stops the Telnet Server. So that no Telnet-Client can connect to the embedded device anymore.

Parameter

UINT16_stx connID
The ID which connection should be stopped.

Return Value

void

Comment

stxTELNET_Start() must have been called before using this function.

3.1.5 stxTELNETServ_Receive (data, size, connID)

Description

This function must be called if data was received on Telnet TCP Port.

Parameter

UINT8_stx FPTR_stx data
Pointer to the received data.

UINT16_stx size
Size of the received data

UINT16_stx connID
The connection ID where the data was received.

Return Value

void

Comment

-

3.1.6 stxTELNETServ_TransportLayerEvent (ev, connid)

Description

This function has to be called from application, dependent on the TCP connection status or when receiving data for Telnet module.

Parameter

TELNET_TLEVENT ev

TELNET_TLE_TXREADY:	TCP is ready to send or has tx'ed a packet
TELNET_TLE_OPEN:	TCP is connected
TELNET_TLE_CLOSE:	TCP was closed

UINT16_stx connid

The connection ID which has received this information.

Return Value

void

Comment

-

3.2 NVT

3.2.1 stxNVT_Tick ( )

Description

This function keeps the NVT alive. It should repeatedly be called by the application, like all the other sevenstax “Tick” functions.

Parameter

void

Return Value

void

Comment

-

3.2.2 stxNVT_Send (sDataTx, size)

Description

This function sends out data to the connected client. Therefore the data will be copied into NVT_Sendbuffer. So the size of data is limited by NVT_SENDBUFFER_SIZE

Parameter

STRING_stx sDataTx

String to send out

UINT16_stx size

The size of the String

Return Value

BOOL_stx

TRUE_stx	if data was sent out
FALSE_stx	if any error occurred

Comment

-

3.2.3 stxNVT_DoEcho (value)

Description

This function configures the NVT module to switch on/off echo for incoming data.

Parameter

BOOL_stx value

TRUE_stx	turn on echo
FALSE_stx	turn off echo

Return Value

-

Comment

-

3.2.4 stxNVT_DoStringMode (value)

Description

This function configures the NVT module to give full strings to the active function or every received char.

Parameter

BOOL_stx value

TRUE_stx	String Mode
FALSE_stx	Char Mode

Return Value

-

Comment

-

4 Implementing User Functions

The Telnet Server provides common functions, which can be used to get information about the embedded device. A main application for the NVT is to implement user functions, which can be used over a Telnet connection by a client. The next chapter will show how to write user functions.

4.1 Developing User functions

The NVT makes it possible to start user functions on the embedded device with the help of a Telnet client. Two data files are needed: *nvtuser.h* and *nvtuser.c*. In the file *nvtuser.c* a function table (sUserFuncTab) is included, where user functions must be registered. A *DemoFunction()* is given to see how to make your own functions. In the *nvtuser.h* file the number of existing User-Functions must be set by changing the number of user functions (*NUMBER_OF_USER_FUNCTIONS)*. This must agree with the actually number of functions in function table.

A User Function must be of type:

UINT32_stx UserFunction (recvdata, info, iError)

recvdata: STRING_stx
if recvdata != 0, then data was given to the User Function

info: NVT_FUNCTION_INFORMATION FPTR_stx
this structure includes some useful pointers to variables (for example a pointer to a counter, which can be used to count up every cycle).

iCounter:	Can be used to count up. When the function is called the first time, the value of the counter is zero.
bCheckCommand:	A String was delivered to function

iError: UINT16_stx
if iError != NVT_NO_ERROR then an error occurred. The meaning of the error-code can be checked in a later chapter.

If the function is called, it receives one of the following error codes:

NVT_NO_ERROR	No error occurred, everything is OK.
NVT_EXIT_REQUEST	The NVT wants to inform the function, that the function will be called only once more. The application can decide if it finish now. Otherwise it will be called with NVT_EXIT_NOW the next time.
NVT_EXIT_NOW	The function is called the last time. It will not be called anymore unless the function will be called a second time due to

activation by typing the command for this function at prompt.

A active function must return one of these values:

FUNCTION_RUNNING	if the function wishes to be called again
FUNCTION_END	if the function does not wish to be called again
SESSION_END	if the function wants to close the Telnet session

5 Notifycodes

5.1 NC_TELNET_OPEN (0, connID)

This event is generated to commit (or recommit) the connection ID on which the Telnet server is listening. This connection ID must be used to submit data or to inform the Telnet module about the TCP connection.

Parameter l *not used and always zero*

Parameter w indicates the ID of the TCP instance, which has been opened.

ReturnValue *not used*

5.2 NC_TELNET_CONNECTED (0, connID)

This event is generated to notify the application, that a client has connected to the Telnet server successful.

Parameter l not used and always zero

Parameter w indicates the ID of the TCP instance.

ReturnValue not used

5.3 NC_TELNET_TX (0, connID)

This event notify the application that the last data was sent successful.

Parameter l not used and always zero

Parameter w indicates the ID of the TCP instance.

ReturnValue not used

5.4 NC_TELNET_RX (0, connID)

This event notify the application that the Telnet server has received data.

Parameter l not used and always zero

Parameter w indicates the ID of the TCP instance, on which this data was received.

ReturnValue not used

5.5 NC_TELNET_OPTION_ON (option, connID)

This event notify the application that the Telnet server option is enabled.

Parameter l the received option which was enabled

Parameter w the connection ID which has data received

ReturnValue not used

5.6 NC_TELNET_OPTION_OFF (option, connID)

This event notify the application that the Telnet server option is disabled.

Parameter l *the* received option which was disabled

Parameter w the connection ID which has data received

ReturnValue not used

5.7 NC_TELNET_ERROR (errCode, connID)

This event notifies the application that an error occurred while receiving the data.

Parameter l error code. (prepared for later usage)

Parameter w the connection ID which has detected an error.

ReturnValue not used.

6 Adjustable Parameters

TELNET parameters are defined in '*telnetdefs.h*' and 'nvtuser.h'. These parameters should properly be adjusted depending on the application requirements. The following list describes them in detail.

Defined in telnetdefs.h:

TELNET_SERVERPORT **default: 23** **valid: 1..n**

TCP-Port for receiving Telnet connections.

Defined in nvt.h:

NVT_SENDBUFFER_SIZE **default: 150** **valid: 150..n**

Limit the payload for sending data via *NVT_Send()*.

NVT_RECVBUFFER_SIZE **default: 150**

Limit the receive buffer for storing data. Should not be changed.

Defined in nvtuser.h:

NVT_NUMBER_OF_USER_FUNCTIONS **valid: 1..n**

The number of user functions MUST agree with the number of user functions in the function table.

NVT_USERNAMES **default:10** **valid: 1..10**

The number of user functions MUST agree with the number of passwords (NVT_PASSWORD)

NVT_PASSWORD **default:10** **valid: 1..10**

The number of user functions MUST agree with the number of passwords (NVT_USERNAMES)

NVT_PROMPT **default:”7stax> ”**

This defines which String will be shown as prompt. The String length must be less than 10 characters.

NVT_GREETINGSCREEN **default:”Welcome to ...”**

This defines which String will be shown at startup as greeting screen. The String length must be less than NVT_SENDBUFFER_SIZE characters.

NVT_USER1 - NVT_USER10 **default:”Guest – Guest9”**

Here the user names can be defined. NVT_USER1 must insert NVT_PASSWORD1 to login correctly.

NVT_PASSWORD1 - NVT_PASSWORD10 **default:”Password – Password9”**

Here the passwords for the users must be defined.

7 Restrictions

sevenstaxTelnet is focused on systems with very small resources, to give them relevant Telnet Server functionality. Therefore – in comparison with a full featured stack implementation of a PC software f.e. - it has some restrictions in usage and capabilities, the user should know:

#	*Restriction*	*Details*
1	Only one Client can connect to the Telnet Server at the same time	For the need of lower resources, the telnet server only support one connection at the same time.
2	The maximal payload size for TCP data packets which should send via Telnet is limited to the sendbuffer size	This is an actually problem and will be fixed in next release. If it is necessary to send bigger packets, the user can change the sendbuffer size (but it must be at least 150 Bytes).

Sevenstax Telnet is object of permanent improvement. Please contact info@sevenstax.de to get more information about new Telnet features given in next release version.

8 Change History

Ver.	*Date*	*by*	*Change description*
1.0	06 Jan. 2008	bbr	Base version
1.1	08 Jan. 2008	jma	Review
1.2	11 Febr. 2008	krz	Review
1.3	12 Febr. 2008	bbr	Restrictions added, notify for Options changed

The information furnished in this document is believed to be accurate and reliable. However, no responsibility is assumed by sevenstax for its use, nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of sevenstax.

This document is an intellectual property of sevenstax GmbH. Unauthorized copying and distribution is prohibited.

Copyright (c) 2008 by sevenstax GmbH